Chapter 15

音樂盒與拼圖

音樂盒與拼圖這兩個原本不相干的兩的主題，因為 Android 專題指導的機會，而有了共同的話題。原來，音樂盒是在文大任教時期所指導一項專題製作的子功能，當時為了突顯觸控螢幕與多媒體之間的互動關係，於是將音樂盒作成一支 App 加以呈現；而拼圖遊戲則是在巨匠指導學員進行專題製作，後來加以改良。[1]

此外，音樂盒與拼圖這兩支 App 都採取了四季的主題畫面與音樂曲目，這些類似的素材促使這兩個 App 小品整合在本章之中。

15.1 音樂盒專題

一般音樂盒多半是將特定數量與曲目固定在音樂盒內，經由某種機制自動或手動的方式播放，而曲目的進行也分成反覆播放以及循序漸進或隨機選曲的方式進行。

本次的專題也雷同，選擇 Vivaldi（韋瓦第）之四季（The Four Seasons）交響曲作為音樂盒的曲目，但只截取片段聊備一格，讀者可以動手自行換成完整版自娛娛人。

[1] 觸控設計之觀念與創意應用 -- 嵌入式系統、人機介面與Android專題實作，碁峰，2010

15.1.1 音樂盒規格簡介

因此，聲光效果之聲音規格部份就以四季定調；而聲光效果的光影部份，也就是圖像方面，在此以兩種意向來呈現四季。首先，以花朵來表現四季，有所謂「春蘭」、「夏荷」、「秋菊」、「冬梅」之說；其次，按著大自然的科學角度，我們知道傾斜 23 度半角的地球繞著太陽所做的公轉，正是影響太陽照射地球不均而形成之四季現象的原因，可以設計相關圖像作為表徵。

而在此兩類圖像上所執行的點擊手勢，則分別定義為音樂播放以及四季切換的功能按鍵。整個畫面以全螢幕的方式來呈現。

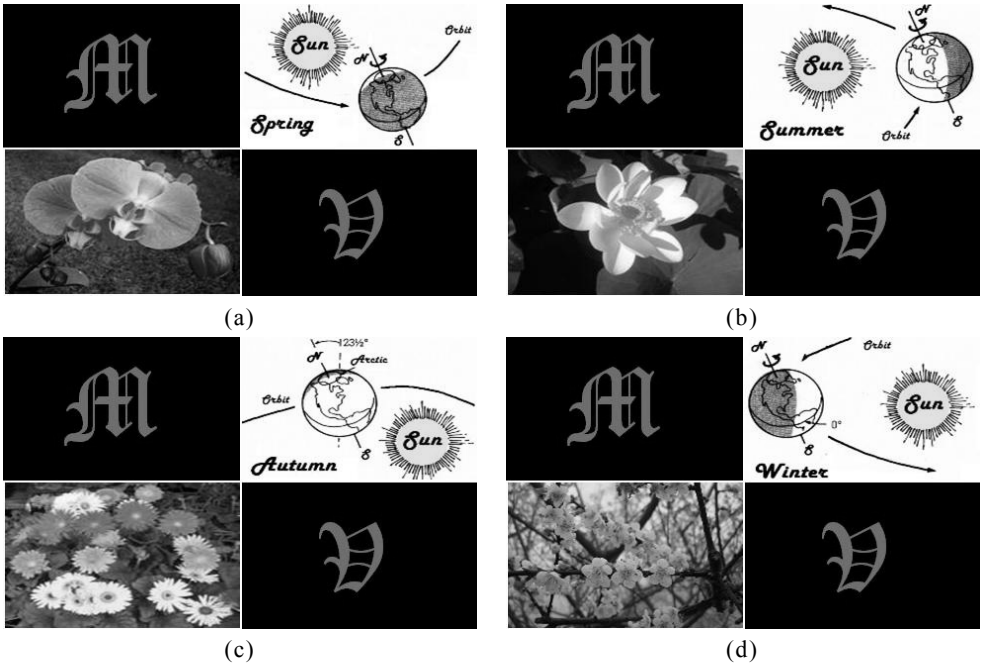


圖 15-1 音樂盒以「四季」為主題之視圖設計，配合 2×2 之矩陣按鍵：(a)春季；(b)夏季；(c)秋季；(d)冬季。

此外，正因為四季的變化有地球公轉的涵意在其中，所以規格也設計靜態與動態兩種行為模式，當樂聲響起的時候，音樂盒上頭 2×2 之矩陣圖像就應該自行轉動起來，彷彿隨著音樂翩翩起舞，讓整個音樂盒的設計更加生動有趣。然而，就在 2×2 的矩陣圖像旋轉之際，也應該要能夠正常接受使用者所執行的點擊手勢，讓音樂盒能夠隨時進行音樂停止或是四季切換的功能。

圖 15-1 就是以此概念所設計的其中四張圖片，分別表達春、夏、秋、冬的感覺；至於圖片中另外兩個字母「m」與「v」英文字首鈕可以再擴充應用。

15.1.2 靜態畫面之視圖設計

如前述，四季主題之內涵意義本身就包含循環不息的概念，本是可以採取動畫效果加以呈現更為精彩好看。然而亦可以採用影像切換（ImageSwitcher）元件之圖片淡進淡出的切換效果，以較低成本達成 UI 之視圖設計；至於其反覆旋轉的視覺效果，則將四季各自作成四張圖片如圖 15-2 所示（四個季節、連同直/橫式，所以共有 4×2×2=16 張圖片需要設計完成）。

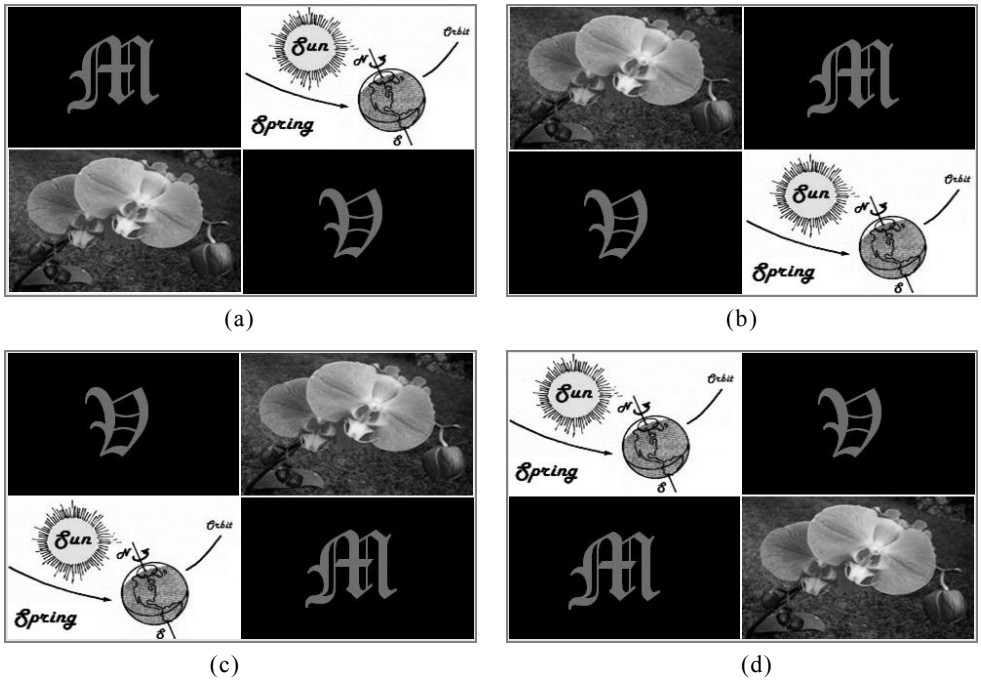


圖 15-2 音樂盒以 2×2 矩陣畫面，從(a)到(d)週而復始顯示，可以呈現旋轉的視圖效果。

程式列表 15-1：Musicbox.java：第 44 至 60 行

```
44 // [iSeason, iCounter]
45 // 看 iSeason = 0~3 就可對應到 四季位置
46 // 看 iCounter = 0~3 就可對應到 M, Flower, V, Track 各自的位置
47 private Integer[][] mImageIdsP = { // 直式
48          {R.drawable.i1_00, R.drawable.i1_01, R.drawable.i1_11,
   R.drawable.i1_10}, // 春
```

```
49     {R.drawable.i2_00, R.drawable.i2_01, R.drawable.i2_11,
   R.drawable.i2_10}, // 夏
50     {R.drawable.i3_00, R.drawable.i3_01, R.drawable.i3_11,
   R.drawable.i3_10}, // 秋
51     {R.drawable.i4_00, R.drawable.i4_01, R.drawable.i4_11,
   R.drawable.i4_10}  // 冬
52    };
53 private Integer[][] mImageIdsL = { // 橫式
54           {R.drawable.i00_1, R.drawable.i01_1, R.drawable.i11_1,
   R.drawable.i10_1}, // 春
55     {R.drawable.i00_2, R.drawable.i01_2, R.drawable.i11_2,
   R.drawable.i10_2}, // 夏
56     {R.drawable.i00_3, R.drawable.i01_3, R.drawable.i11_3,
   R.drawable.i10_3}, // 秋
57     {R.drawable.i00_4, R.drawable.i01_4, R.drawable.i11_4,
   R.drawable.i10_4}  // 冬
58    };
59
60 private int [] mSongIds = {R.raw.spring, R.raw.summer, R.raw.autumn,
   R.raw.winter};
```

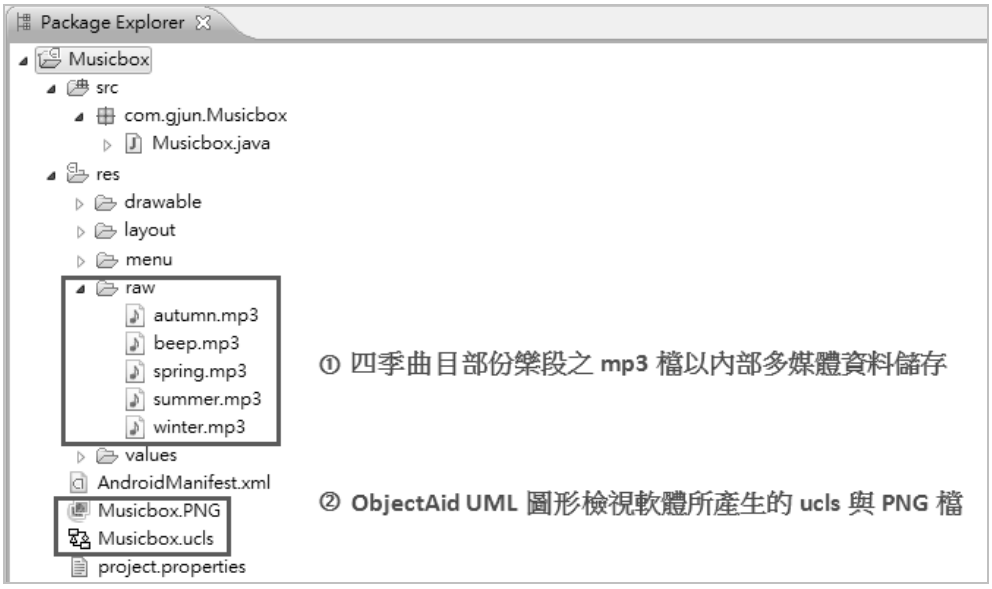


圖 15-3 四季曲目之 mp3 檔和 UML 檢視檔於專案中的擺放位置。

15.1.3 動態聲光之元件使用

確認以 ImageSwitcher 作為視圖顯示的機制之後，就要實作 ViewSwitcher.ViewFactory 介面，因為有一個抽象方法：

```
public View makeView();
```

是在 ImageSwitcher 物件執行 setFactory(this);所需要用到的（如程式列表 15-2 第 108 行）。

這時，動態的光影效果就交給 Handler+Runnable+Timer+ TimerTask 之背景計時器元件組合來執行（如列表 15-2 第 69 至 86 行、183 至 185 行所示）。這部份的組合運作原理已經在前面篇章討論過，就不再贅述。

至於動態的聲音效果，為何沒有利用這四個元件的組合？是因為一旦音樂播放下去就一直循環 looping 下去（如列表 15-2 第 180 行和第 187 至 196 行所示），直到手指按下 ImageSwitcher 視圖上的 2x2 圖片中的花的部份。

最後，關於音量的部份，可以藉由 AudioManager 來取得相關音量資訊：

```
am = (AudioManager)getSystemService(Context.AUDIO_SERVICE);
```

而這個版本的音樂盒作法較為簡易，就是取得所持手機的最大音量值直接作為播放的音量指標（如列表 15-2 第 129 至 131 行和第 192 行所示）。

程式列表 15-2：Musicbox.java

```
30 public class Musicbox extends Activity implements
   ViewSwitcher.ViewFactory {
     ...
35   private ImageSwitcher mImgSwitcher;
36   private AudioManager am;
37   private private curVol, maxVol;
     ...
69   private MediaPlayer mMediaPlayer = null;
70   private Handler mHandler  = new Handler();
71   private Runnable runDancing = new Runnable() {
72      public void run() {
73         dancing();
74      }
75   };
```

```
76    private Timer timer;
77    class Task extends TimerTask{
78       public void run(){
79          execute();
80        }
81       public synchronized void execute() {
82          mHandler.removeCallbacks(runDancing);
83          mHandler.post(runDancing);
84        }
85    };// Task
86    public Task task = new Task();
87
88    private synchronized void dancing() {
89       iCounter = ((++iCounter)+4)%4;
90       updateScreen();
91    }
92
93    @Override
94    protected void onCreate(Bundle savedInstanceState) {
         ...
107      mImgSwitcher = (ImageSwitcher) findViewById(R.id.imgswitcher);
108      mImgSwitcher.setFactory(this);
109      mImgSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
110                  android.R.anim.fade_in));
111      mImgSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
112                  android.R.anim.fade_out));
113
114      mImgSwitcher.setPadding(0, 0, 0, 0);
115      if (width > height) //---landscape mode---
116
     mImgSwitcher.setImageResource(mImageIdsL[iSeason][iCounter]);//[i
     Counter+iSeason*4]);
117      else //---portrait mode---
118
     mImgSwitcher.setImageResource(mImageIdsP[iSeason][iCounter]);//mI
     mageIdsP[iCounter+iSeason*4]);
119
     ...
129      am = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
130      maxVol = am.getStreamMaxVolume(AudioManager.STREAM_RING);
131      curVol = maxVol;
132   }
     ...
172   private synchronized void updateScreen() {
173      if (width > height)
```

```
174   mImgSwitcher.setImageResource(mImageIdsL[iSeason][iCounter]);
175      else
176        mImgSwitcher.setImageResource(mImageIdsP[iSeason][iCounter]);
177   }
178   private void dancingAndSing() { // 螢幕跳舞
179      indexOfSong = iSeason;
180      playAudio(indexOfSong, curVol, curVol);
181
182      bDancing = true;
183      timer = new Timer();
184      task = new Task();
185      timer.scheduleAtFixedRate(task, 1000 ,2000); // 2秒後動作，每1
   秒作一次
186   }
187   private void playAudio(int indexOfSong, int curVol, int maxVol) {
188      try {
189         mMediaPlayer = MediaPlayer.create(this,
   mSongIds[indexOfSong]);
190         mMediaPlayer.start();
191         mMediaPlayer.setLooping(true);
192         mMediaPlayer.setVolume(curVol*1.0f, curVol*1.0f);
193      } catch (Exception e) {
194         System.out.println( e.getMessage().toString());
195      }
196   }
      ...
240   public View makeView() {
241      ImageView i = new ImageView(this);
242      i.setScaleType(ImageView.ScaleType.FIT_XY);
243      i.setLayoutParams(new
   ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,
244            LayoutParams.FILL_PARENT));
245      return i;
246    }
247 }
```

15.1.4 動態點擊之聲光切換

如前述，音樂之停止與播放在這個音樂盒的設計是藉由 Touch 的動作來觸發，因此需要整合 OnTouchListener；不採用 OnClickListener 的原因是因為還需要判斷所點擊到的位置是 2x2 中的哪一塊區域？因此需要將所觸控的點

座標換算成 rowButton 與 colButton，然而據以進行 MVC 的 Model 部份，如程式列表 15-3 之第 120 至 128 行所示。

進到第 133 至 168 行的 `click(int rowButton, int colButton)`函式方法之後，只要是 `mMediaPlayer.isPlaying()`返回值為真，無論按到的是 2x2 的哪一區塊，都要先將 `mMediaPlayer` 停下來，因為要不就是換歌，要不就是要執行其他 AP。相關執行流程如圖 15-4 所示。

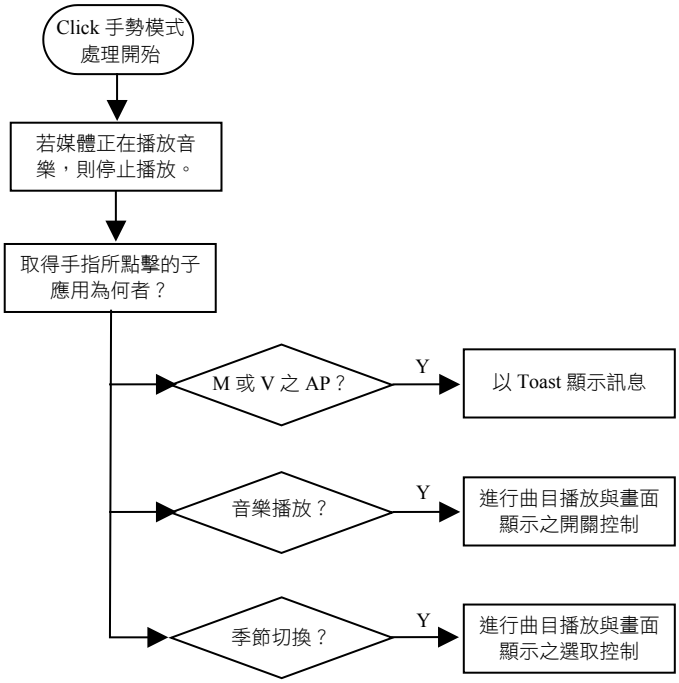


圖 15-4 程式列表 15-3 之 2×2 四個按鍵區塊點擊後的執行流程。

程式列表 15-3：Musicbox.java

```
120         mImgSwitcher.setOnTouchListener(new OnTouchListener() {
121          @Override
122          public boolean onTouch(View v, MotionEvent event) {
123               rowButton = (int) ((event.getY() / (v.getHeight()/2)));
124               colButton = (int) ((event.getX() / (v.getWidth()/2)));
125               if(event.getAction() == MotionEvent.ACTION_UP)
126                    click(rowButton, colButton);
127               return true;
128          }});
     ...
```

```
133 private void click(int rowButton, int colButton){
134
135     bJustPlaying = false;
136      if(mMediaPlayer!=null) {
137        if(mMediaPlayer.isPlaying()) {
138           bDancing = false;
139           timer.cancel();
140
141           bJustPlaying = true;
142           mMediaPlayer.stop();
143           mMediaPlayer.release();
144           mMediaPlayer = null;
145        }
146      }
147      int iPos = whichPOS(rowButton, colButton, iCounter);
148      System.out.println(", "+iSeason);
149      switch(iPos) {
150           case POS_M:
151                Toast.makeText(this, "Reserved for App M.",
    Toast.LENGTH_LONG).show();
152                break;
153           case POS_FLOWER:
154                if(!bJustPlaying)
155                     dancingAndSing();
156                break;
157           case POS_V:
158                Toast.makeText(this, "Reserved for App V.",
    Toast.LENGTH_LONG).show();
159                break;
160           case POS_TRACK:
161                iSeason++;
162                iSeason%=4;
163                updateScreen();
164                if(bJustPlaying)
165                     dancingAndSing();
166                break;
167     }
168 }
169 private int whichPOS(int row, int col, int counter) {
170    return fourRounds[ rowcol[row][col] ][iCounter];
171 }
```

圖 15-4 中有一項關鍵步驟稱為『取得手指所點擊的子應用為何者？』，執行位於列表 15-3 的第 147 行以及第 170 行的內容：

- `int iPos = whichPOS(rowButton, colButton, iCounter);`
- `return fourRounds[ rowcol[row][col] ][iCounter];`

其中以所點擊的 row、column 位置，配合 iCounter，就能取得所對應之子應用！但它的原理是從 row、column 值對應到圖片之 2×2 的四個角落位置：

```
private int [][] rowcol = { {LEFT_TOP, RIGHT_TOP}, {LEFT_BUTTOM,
RIGHT_BUTTOM} };
```

再配合 iCounter 的值，就能在 fourRounds 二維陣列中，索引到子應用的位置！

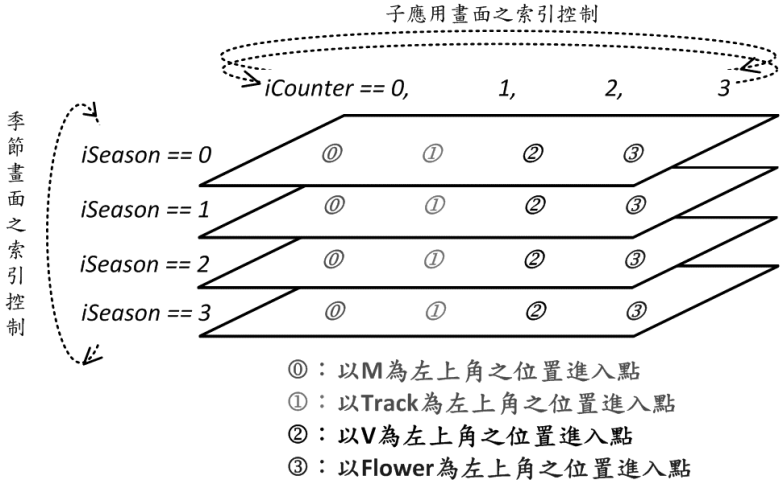


圖 15-5 針對程式列表 15-1 之 2×2 四個按鍵區塊於視圖位置的索引控制設計。

至於為什麼 fourRounds 二維陣列可以與直式的 mImageIdsP[][]、或是橫式的 mImageIdsL[][]二維陣列之資源 ID 相對應，則可從圖 15-5 一窺究竟。

圖 15-5 可以看到一個稱為『子應用畫面之索引』的 iCounter 變數，可以遞增或遞減的方式來作循環控制；另外一個稱為『季節畫面之索引』的 iSeason 變數則會以 0、1、2、3、0…之遞增式循環來作控制，其順序能和春、夏、秋、冬四季的次序吻合。值得注意的是，iSeason 並不會影響子應用之相對位置。

15.2 拼圖遊戲

遊戲的應用在手持式裝置上可說是不可或缺的產品功能，不論是為了寓教於樂，或純粹只是打發時間。而這種情況，在觸控式智慧型手機或平板電腦愈發普及的現今則更有過之而無不及。因為這類裝置的作業系統如 Android，往往將各式感測器如方位感測器、重力加速度感測器等也納入標準配備，或是已提供 SDK 之標準 API 就等著廠商的硬體來支援與使用，在在推動著遊戲的普遍化與精緻化。

遊戲最重要的質素當然在於主題的趣味與良好的操控及畫面。其次，計分、評比方式以及功能設定與介紹等「配角」介面也常需要擔任綠葉的功能，好襯托出遊戲如紅花般地令人愛不釋手！筆者於本節透過拼圖遊戲的設計，一方面介紹遊戲之主配角功能的元素，另一方面提出創意手勢的應用作為對照。

本節在 Android 的技術方面包括 ImageView 切割與重組、AlertDialog 內元件之顯示與控制、動畫展示、動態更新頁面、彩色轉灰階 API 設計、SQLite 函式檢索技巧、滑動手勢取代點擊等等，材料有趣且實用。

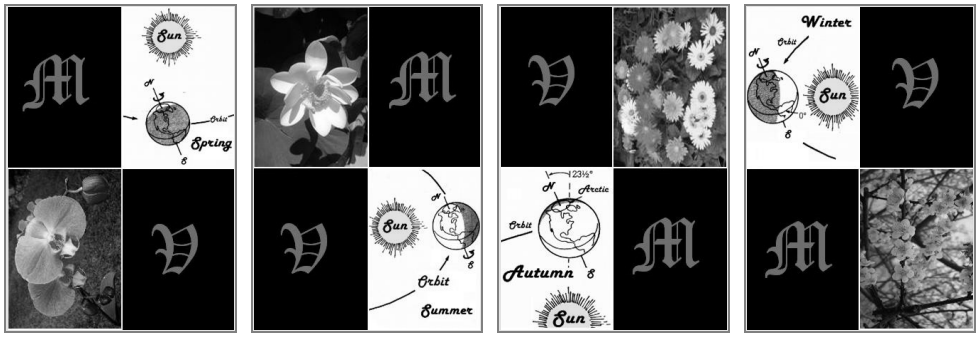
15.2.1 拼圖遊戲之功能介紹

將圖片以二維陣列方式切開重組的方式進行拼圖的玩法，其實不算新創，而是許多紙片拼圖業者一直存在的行銷手法之一，特別是由客戶提供自己心愛的照片所加工而成的拼圖表框，更是常見的裝飾擺設品。在本章的專案中，筆者以前一小節的四季音樂盒之圖片作為拼圖的底圖，算是充分利用。

以現今手機的屏幕大小來看，適合分成難（5×5）、中（4×4）、易（3×3）三種等級進行，但若以平板電腦的尺寸來分析，則又另當別論。既然遊戲已採取分級進行方式，則評分也應分級記錄。筆者以 Android 所提供為每個應用程式所特有之 SQLite 資料庫，在其中建立資料表，表格中存放每一級前 10 名的玩家名稱與相關成績資訊。

而拼圖所用的四張底圖則運用 Android 所提供繪製圖像到圖像顯示物件（ImageView）上，將圖像放進圖像顯示物件中，加以事先播放，次序如圖 15-6 所示，讓玩家事先了解「劇情」。

程式部份讀者可參考列表 15-4 了解如何利用資源宣告 animation-list 標籤項目，以及列表 15-5 如何利用 AnimationDrawable 物件配合 Handler 與 Runnable 物件進行動畫播放。



(a) (b) (c) (d)

圖 15-6 拼圖遊戲開始前，循序播放四種可能出現之主題的最終完成畫面預覽：(a)春季；(b)夏季；(c)秋季；(d)冬季。

程式列表 15-4：four_seasons.xml

```
01      <?xml version="1.0" encoding="UTF-8"?>
02      <animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
            android:oneshot="true">
03          <item android:drawable="@drawable/i1_00"
    android:duration="1000"/>
04          <item android:drawable="@drawable/i2_01"
    android:duration="1000"/>
05          <item android:drawable="@drawable/i3_11"
    android:duration="1000"/>
06          <item android:drawable="@drawable/i4_10"
    android:duration="1000"/>
07  </animation-list>
```

程式列表 15-5：Animate.java

```
12  public class Animate extends Activity {
13      private Handler mHandler;
14      private ImageView img;
15      private AnimationDrawable ad;
16      @Override
17      public void onCreate(Bundle savedInstanceState) {
18          super.onCreate(savedInstanceState);
19          requestWindowFeature(Window.FEATURE_NO_TITLE); // 去除標題
```

```
20        setContentView(R.layout.animate);              // 設定版面
21        img = (ImageView)findViewById(R.id.Image01);
22        img.setBackgroundResource(R.anim.four_seasons); // 以 xml 動畫設
    定 ImageView
23        img.setScaleType(ScaleType.FIT_XY);           // 填滿畫面
24        ad=(AnimationDrawable)img.getBackground();     // 取出動畫
    Drawable
25
26        mHandler = new Handler();
27        mHandler.postDelayed(anim, 10);    // 10 毫秒之後啟動動畫
28        mHandler.postDelayed(update,4000);  // 四秒之後切換視窗
29     }
30     private Runnable anim = new Runnable() {
31        public void run() {
32           ad.start();                    // 啟動動畫 Drawable
33        }
34     };
35     private Runnable update = new Runnable() {
36        public void run() {
37           mHandler.removeCallbacks(update);
38           Intent intent = new Intent();
39           PlayPuzzle.isPlaying = true;
40           intent.setClass(Animate.this,PlayPuzzle.class);
41           startActivity(intent);
42           finish();                      // 關閉此動畫視窗
43        }
44     };
45  }
```

15.2.2 選單設計與拼圖製作

由於微軟 Office 軟體就有提供拼圖圖案，剛好四塊拼圖符合筆者在此規劃的四項功能：Playing（遊戲開始）、Setting（喜好設定）、Ranking（成績排行）、以及 Helping（功能說明），選單操作畫面如圖 15-7 所示。玩家可以用手指在上面滑動，直到離手時才會觸發執行相對的功能。

從圖 15-7 可以看出畫面中四塊拼圖按鈕的顏色有深淺的變化，是因為在程式列表 15-6 內對於相對的 ImageView 物件以 setImageResource()進行背景圖的切換所導致的視覺效果！

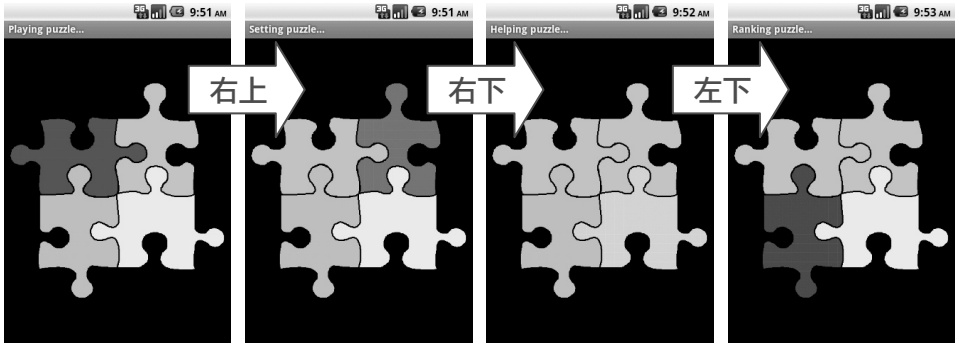


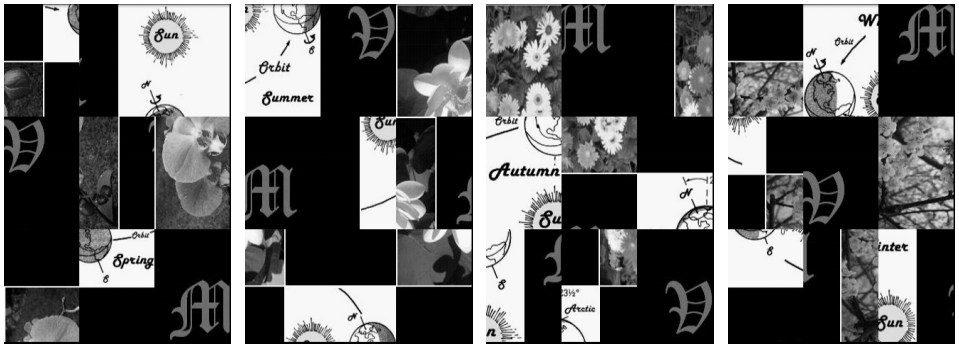
圖 15-7 拼圖選單之功能選取範例畫面：(a)左上角 Playing 功能；(b)右上角 Setting 功能；(c)右下角 Helping 功能；(d)左下角 Ranking 功能。

程式列表 15-6：MainMenu.java：第 24、38 至 67 行

```
24 iv = (ImageView) findViewById(R.id.ImageView01);
   ...
38 public void imageFocused(float x, float y, int w, int h) {
39      if(x>=0 && x<w/2 && y>=0 && y<h/2) {
40           setTitle("Playing puzzle...");
41          iv.setImageResource(R.drawable.icon_big_r);
42      }
43      else if(x<w && x>=w/2 && y>=0 && y<h/2)  {
44          setTitle("Setting puzzle...");
45           iv.setImageResource(R.drawable.icon_big_g);
46      }
47      else if(x>=0 && x<w/2 && y<h && y>=h/2)  {
48          setTitle("Ranking puzzle...");
49           iv.setImageResource(R.drawable.icon_big_b);
50      }
51      else if(x<w && x>=w/2 && y<h && y>=h/2)  {
52          setTitle("Helping puzzle...");
53           iv.setImageResource(R.drawable.icon_big_y);
54      }
55 }
56 @Override
57 public boolean onTouch(View v, MotionEvent event) {
58      // TODO Auto-generated method stub
59      float x = event.getX();//讀X座標
60      float y = event.getY();//讀Y座標
61      int w = v.getWidth();
62      int h = v.getHeight();
63      switch(event.getAction()) {
```

```
64          case MotionEvent.ACTION_DOWN:
65          case MotionEvent.ACTION_MOVE:
66              imageFocused(x, y, w, h);
67              break;
        ...
```

至於拼圖製作的部份，如圖 15-6 所預覽的四個季節主題圖，以隨機切割重組的方式加以製作，可分別對應到如圖 15-8 的 a、b、c、d 四個圖面，再由玩家第一次觸碰屏幕之後開始計時。程式列表 15-7 顯示如何將原始圖片分割，它先藉由 produceRandomArr()函式以亂數方式將 n×n 張圖片重新編號，再運用 preparePuzzle()函式依照所重編的次序，將圖片以 Bitmap.createBitmap()函式切成小塊再放入新的次序中。



(a) (b) (c) (d)

圖 15-8 拼圖遊戲之四種主題畫面隨機出現：(a)春季；(b)夏季；(c)秋季；(d)冬季。

程式列表 15-7：PlayPuzzle.java：第 167 至 199 行

```
167 // 準備隨機亂數陣列
168 private void produceRandomArr() {
169     int imgLen = numPerLine * numPerLine;
170     ArrayList<Integer> org = new ArrayList<Integer>();
171     for (int i = 0; i < imgLen; i++)
172         org.add(i, i);
173     int len = imgLen;
174     for (int i=imgLen; i>=1; i--) {
175         int iShuffle = (int) (Math.random() * i);;
176         randomIdxImg[i-1] = org.remove(iShuffle);
177     }
178 }
179 // 以隨機亂數陣列安排拼圖圖片
180 private void preparePuzzle() {
```

```
181     Matrix matrix = new Matrix();
182     matrix.postScale(scaleWidth, scaleHeight);
183
184    Bitmap pieces[][] = new Bitmap[numPerLine][numPerLine];
185     for (int i = 0; i < numPerLine; i++)
186          for (int j = 0; j < numPerLine; j++)
187               pieces[i][j] = Bitmap.createBitmap(map, blockWidth * j,
   blockHeight * i,
188                         blockWidth, blockHeight, matrix, true);
189
190     for (int i = 0; i < randomIdxImg.length; i++) {
191          int xS = randomIdxImg[i] / numPerLine;
192          int yS = randomIdxImg[i] % numPerLine;
193
194          int xD = i / numPerLine;
195          int yD = i % numPerLine;
196
197          randomBmps[xD][yD] = pieces[xS][yS];
198     }
199 }
```

程式列表 15-8 則顯示如何將切成小塊的圖片作成 View 物件，讓其他程式可呼叫第 30 到 66 行所提供的 setContentView()函式來設定它的視圖。

程式列表 15-8：ImageReorder.java：第 24、27 至 66 行

```
24 public ImageReorder(Context context, ImageView[][] views) {
       ...
27     this.views = views;
28     setContentView();
29 }
30 private void setContentView() {
31     linear = new LinearLayout(mContext);
32     linear.setOrientation(LinearLayout.VERTICAL);
33     for (int i = 0; i < views.length; i++) {
34          ll = new LinearLayout(mContext);
35          ll.setOrientation(LinearLayout.HORIZONTAL);
36          for (int j = 0; j < views[i].length; j++) {
37               ImageView img = (ImageView)views[i][j];
38               // 將切割的拼圖儲存到 sdcard
39               Drawable da = img.getDrawable();
40               Bitmap image_saved = ((BitmapDrawable)da).getBitmap();
41              FileOutputStream fOut;
42               try {
43                    String tmp = "/sdcard/img_"+i+"_"+j+".jpg";
```

```
44                    fOut = new FileOutputStream(tmp);
45
      image_saved.compress(Bitmap.CompressFormat.JPEG,100,fOut);
46                    System.out.println(tmp);
47                    try {
48                         fOut.flush();
49                         fOut.close();
50                    } catch (IOException e) {
51                         // TODO Auto-generated catch block
52                         e.printStackTrace();
53                    }
54               } catch (FileNotFoundException e) {
55                    // TODO Auto-generated catch block
56                    e.printStackTrace();
57               }
58               if(img != null)
59                    ll.addView(img,new LayoutParams(
60                              LayoutParams.WRAP_CONTENT,
61                              LayoutParams.WRAP_CONTENT));
62          }
63          linear.addView(ll);
64     }
65     this.addView(linear);
66 }
```

讀者可以看到也可運用列表 15-8 第 39 到 57 行的程式片段，特別是從 Drawable 利用 getBitmap()轉到 Bitmap、再利用 compress()轉到 JPEG 或 PNG 檔案格式儲存起來。

15.2.3 排行資料庫與最愛設定

遊戲程式很重要的一項功能是評分與排行，而排名取前十名存入資料庫中隨時可以查閱，則也是一種常見的激勵手段，就是所謂的排行榜，因為爭取入榜是達到成就感的動機之一。圖 15-9 顯示以「秒」為單位進行排行，時間愈短當然排行愈前面。圖 15-9d 所包含的資訊有流水號（id）、姓名（name）、闖關時間（time）、完成時間（created）與困難度（difficulty）。

程式列表 15-9：RankDB.java：第 28 至 34 行

```
28 public interface UserSchema {
29      String TABLE_NAME = "Players";
30      String ID = "_id";
31      String NAME = "name";
32      String TIME = "time";
33      String CREATED = "created";
34      String DIFFICULTY = "difficulty"; }
```

圖 15-9　拼圖遊戲之排行資料庫相關主要操作：(a)查詢前十名；(b)新增一筆前十名資料；(c)新增完畢自動更新畫面；(d)點擊查看資料細節，並決定是否刪除。

程式列表 15-10：RankDB.java：第 51 至 76 行

```
51 helper = new DBConnection(this);
52 final SQLiteDatabase db = helper.getReadableDatabase();
53 c1 = db.query("Players", null , "difficulty = '"+level+"'", null, null,
   null, UserSchema.TIME, "10");
54 startManagingCursor(c1);
55 extras = getIntent().getExtras();
56 if (extras != null && fromOutside) {
57    sTime = (String) extras.getCharSequence("TIME");
58    sCreated = (String) extras.getCharSequence("CREATED");
59    sDifficulty = (String) extras.getCharSequence("DIFFICULTY");
60    LayoutInflater factory = LayoutInflater.from(RankDB.this);
61    final View textEntryView =
   factory.inflate(R.layout.alert_dialog_text_entry, null);
62    new AlertDialog.Builder(RankDB.this)
63      .setTitle(R.string.input_hint)
64      .setView(textEntryView)
65      .setPositiveButton(R.string.dialog_ok, new
   DialogInterface.OnClickListener() {
66         public void onClick(DialogInterface dialog, int whichButton) {
67            extras = null;
68            EditText et = (EditText)
   textEntryView.findViewById(R.id.username_edit);
69            if(et.getText().length()!=0)
70                 sName = et.getText().toString();
71            else
72                 sName = "Player_" + new Date().toLocaleString();
73            insert();
74         }
75    }).show();
76 }
```

程式列表 15-9 定義資料表的欄位名稱，可於建立 SQLite 資料庫時加以運用，其欄位的內容可以對照圖 15-9d 來看。程式列表 15-10 則有以下重點：

連接資料庫、建立資料表（第 51 到 52 行）：利用 SQLiteOpenHelper 類別，建立資料庫（名稱：PuzzleScores）與資料表（名稱：UserSchema.TABLE_NAME，即 Players）。※注意：這個套件的用法與第八章所介紹的資料庫用法不同，而是與第 12 章相同，讀者可以對照比較其差異。

1. 以闖關時間為遞增條件排序資料表（第 53 行）：欄位名稱 UserSchema.TIME，即 time，並只取前十名顯示。